Information Processing: Proceedings of the Int'l Conf on Info Proc. UNESCO, Paris, June 15-20, 1959

B | Time sharing in large fast computers

By C. Strachey, National Research Development Corporation, London (UK)

Time sharing, in the sense of causing the main computer to interrupt its program to perform the arithmetic and control operations required by external or peripheral equipment, has been used on a limited scale for a long time. This paper explores the possibility of applying time sharing to a large fast computer on a very extensive scale.

The first main advantage is that the distinction between on-line and off-line equipment largely disappears—the local buffer stores and control units of off-line or autonomous units are no longer needed. The second main advantage is that the efficiency of using the machine can be much improved by having two or more programs in the machine at the same time and arranging that, effectively, the machine is never idle between programs. Other advantages are that the checking of programs should be cheaper and the machine maintenance time may be reduced.

Time sharing on this scale means that there would be various programs and pieces of peripheral equipment competing both for the use of control (time sharing) and also for the use of the store (space sharing). The paper makes definite proposals for a method to reconcile these demands and to prevent interference by one program with another even if one of the programs is incorrect and "runs wild." These involve the use of a permanent main steering program, known as the "Director" and special instructions for transferring the control and for isolating sections of the store which are only available to the director.

Répartition temporelle dans les grandes calculatrices rapides. Le «partage du temps» qui consiste à interrompre le programme de la calculatrice proprement dite pour effectuer les opérations arithmétiques et de contrôle qu'exigent les éléments externes ou périphériques, se pratique depuis longtemps dans une certaine mesure. L'auteur recherche s'il est possible d'étendre cette notion à une grande calculatrice rapide.

Le principal avantage de la méthode est d'abolir presque totalement la distinction entre éléments connectés et éléments périphériques, les mémoires tampons et les unités de commande des éléments périphériques devenant inutiles. Son deuxième grand avantage vient de ce qu'elle permet d'améliorer considérablement le rendement de la machine en y introduisant simultanément deux programmes ou plus et en faisant en sorte qu'elle ne reste jamais inactive. En outre, cette méthode devrait abaisser le prix de revient de la mise au point des programmes et réduirait sans doute la durée des opérations d'entretien.

A cette échelle, la programmation simultanée a pour effet de mettre plusieurs programmes et plusieurs éléments périphériques en concurrence tant pour l'utilisation de l'organe de commande (partage du temps) que pour celle des mémoires (partage de l'espace). L'auteur formule un certain nombre de propositions précises en vue de la mise au point d'une méthode permettant de concilier ces exigences et d'empêcher un programme de perturber l'exécution d'un autre, même s'il est défectueux et «déraille». Sa solution suppose l'emploi d'un programme principal permanent dit programme «directeur» et d'instructions spéciales pour faire passer l'organe de commande d'un programme à l'autre et pour isoler les mémoires auquels le programme directeur est seul à avoir accès.

Zeitverteilung in großen und schnellen Rechenautomaten. In beschränktem Umfang hat man schon seit langer Zeit eine Zeitverteilung in dem Sinnc durchgeführt, daß man den Hauptrechenautomaten sein Programm unterbrechen ließ, um Rechen- und Steueroperationen für die zugehörigen äußeren Geräte auszuführen. Dieser Bericht untersucht auf einer breiten Basis die Anwendungsmöglichkeiten von Zeitverteilungsverfahren auf große und schnelle Rechenmaschinen.

Der erste Hauptvorteil besteht darin, daß die Unterscheidung zwischen angeschlossenen und selbständigen Einheiten weitgehend verschwindet. Die örtlichen Pufferspeicher und Kommandowerke der äußeren oder selbständigen Einheiten werden nicht mehr benötigt. Weiterhin kann der Wirkungsgrad beim Betrieb der Maschine sehr gesteigert werden, indem man zwei oder mehr Programme zur gleichen Zeit in der Maschine hat und es so einrichtet, daß die Maschine zwischen den Programmen niemals leerläuft. Andere Vorteile liegen in der Möglichkeit des billigen Prüfens von Programmen und der Verkürzung der Wartungszeit der Maschine.

Zeitverteilung bedeutet auf dieser Ebene, daß verschiedene Programme und verschiedene angeschlossene äußere Geräte für die Benutzung der Steuerung (Zeitverteilung) und auch für die Benutzung des Speichers (Platzverteilung) in Konkurrenz stehen. Die Arbeit macht bestimmte Vorschläge für eine Methode, um diese Ansprüche miteinander zu vereinbaren und das Dazwischenkommen eines Programmes in ein anderes zu vermeiden, selbst wenn eines der Programme falsch ist und „wild" läuft. Diese Vorschläge enthalten den Gebrauch eines bleibenden Haupt-Steuerprogrammes („director" genannt) und die Verwendung besonderer, nur dem „director" zugänglicher Befehle zur Übernahme der Steuerung und zum Isolieren von Teilen des Speichers.

Совмещение по времени в больших быстродействующих вычислительных машинах. Совмещение по времени, в том смысле, что оно заставляет вычислительную машину прерывать выполняемую ею программу, для того, чтобы выполнить арифметические действия и операции управления, как того требуют внешнее оборудование или оборудование ввода-вывода, давно уже используется в ограниченном масштабе. В докладе исследуется возможность применения совмещения по времени в большой быстродействующей вычислительной машине в очень широком масштабе. Первым основным преимуществом является то, что различие между внешним оборудованием и оборудованием самой машины в значительной степени исчезает — внешние местные буферные запоминающие устройства управления или автономные устройства больше не нужны. Второе основное преимущество заключается в том, что эффективность использования машины может быть значительно улучшена путем одновременного использования двух или более программ в машине и такой организации работы машины, чтобы она никогда не простаивала между программами. Из других преимуществ следует упомянуть удешевление проверки программ и сокращение времени обслуживания.

Совмещение по времени в этом масштабе означает, что различие программы и отдельные части оборудования ввода-вывода будут претендовать на использование управления (совмещение по времени), а также и на использование запоминающего устройства (совмещение по пространству). В докладе даются определенные предложения по методу согласования этих требований и предупреждения вмешательства

одной программы в выполнение другой программы, даже если одна из программ неправильна и выполняется не по графику. Все это требует применения постоянной главной управляющей программы, известной под названием «директор» и специальных команд для передачи управления и для изоляции секций запоминающего устройства, которые доступны только для «директора».

La distribución del tiempo en las grandes calculadoras rápidas. La distribución del tiempo, para lograr la interrupción del programa que está efectuando la calculadora principal para realizar operaciones aritméticas y de control requeridas por la instalación externa y periférica, se ha empleado durante mucho tiempo en escala muy limitada. En este estudio se analiza la posibilidad de aplicar esa distribución del tiempo a una gran calculadora rápida en una escala amplísima.
La primera ventaja importante de este procedimiento consiste en la desaparición casi total de la distinción entre el dispositivo de la línea interior y el de la línea exterior, no necesitándose ya, como consecuencia, los almacenes intermediarios y las unidades de control de la línea exterior, así como las unidades autónomas.
La segunda ventaja consiste en la posibilidad de mejorar grandemente la eficacia en el uso de la máquina gracias al funcionamiento simultáneo en la misma de dos o más programas, y a la seguridad efectiva de que la máquina no se halla nunca inactiva entre un programa y otro. Existen aún otras ventajas, como el abaratamiento de las operaciones de verificación del programa y la reducción del tiempo de mantenimiento de la calculadora.
La distribución del tiempo, realizada a esta escala, significa que varios programas y piezas del dispositivo periférico competirán para utilizar el control (distribución del tiempo) y también para el empleo del almacén (distribución del espacio). En el artículo se formulan propuestas concretas en relación con un método destinado a hacer compatibles tales exigencias, previniendo, a la vez, la interferencia entre un programa y otro, aun en el caso de que uno de ellos sea incorrecto y funcione "desordenadamente". Esto implica el empleo de un programa permanente de gobierno central, denominado "Director", así como de instrucciones especiales para la transferencia del control y el aislamiento de aquellas secciones del almacén que sólo son utilizables por el citado programa director.

1. Introduction

One of the most difficult problems which face the designer of an electronic computing machine is the serious imbalance which exists between the speed of the electro-mechanical equipment used for input and output, and the speed of operation of the wholly electronic internal arithmetic and control circuits.
The solution adopted by most of the fast large scale computers in use today is to have wholly or partly autonomous units, each with its own buffer store.
The disadvantage of this plan is largely an economic one, for as the demands on the system have grown, so the autonomous units have become more and more complex and expensive, until we have reached the position where the peripheral equipment is more expensive than the main computer.
The difficulty is that the logical complexity of the operations required of the autonomous units is quite great. It is true that the rate at which these operations have to be performed is not very high, but unfortunately the cost of a special purpose computer is determined more by its logical complexity than its speed, so that it is not much cheaper to build a slow unit. In fact one of the main economic reasons for building faster computers is that operation for operation they are a great deal cheaper than slow ones.
A concrete example may, perhaps, help to define our ideas somewhat more clearly. I give below the specification of a possible set of functions (or subroutines) for dealing with magnetic tape. (I have purposely oversimplified this example in some respects.)

2. Magnetic tape operations

2.1. *Properties of magnetic tape system*

Information transferred in blocks of fixed length
Each character on tape has a parity check bit
Each word in the main store has a parity check bit
Each block on tape has a check sum
Reading and writing can be done in one direction only
There is a reading head immediately following the writing head to allow a writing check. The separation between these two heads cannot be guaranteed closely so that their timing is not connected.

2.2. *Facilities to be provided*

a) Four orders:
- I) Read a block from tape unit x into the main store starting at address y
- II) Write a block to tape unit x from the main store starting at address y
- III) Skip y blocks on tape unit x
- IV) Search tape unit x for a block starting with a word identical to the word in main store address y.

b) Checks as follows:
- Parity check on characters, sum check on blocks for reading, skipping and searching (orders I, III and IV)
- For writing (II), insert sum check and perform word by word read back check.
- Automatic re-read or re-write on check failure up to a maximum of 4 times, accumulating tallies of errors for each tape unit.
- Special alarm after 4 re-runs.

c) Lock outs:
- During reading or writing the main computer is to be locked out from the single block of store starting at address y until the magnetic tape operation has been completed and successfully checked.

It will be seen that there is a considerable logical complexity in this specification. It is rather improbable that any autonomous buffer unit would be able to fulfill all these requirements; almost certainly the designated main store address y would be confined to the buffer store so that the magnetic tape order would have to be associated with another transfer order between the main store and the buffer. Even this degree of complexity is not as much as may be required. If the requirement of a check sum for each block is removed, it becomes possible to use each word of the main store as soon as it has been dealt with (instead of waiting till the end of the block). This allows much simpler tape copying schemes, but introduces some very severe complications if an error is detected (by parity or read back) in the middle of a block. This sort of situation is beyond the power of a reasonably simple special purpose system, but is relatively easy to programme if the full facilities of the central computer are available.

3. Time sharing solution

We are thus led to consider the possibilities of "borrowing" the central computer for a short time in order to control the magnetic tape operations. This will only be practicable if the computer is a very fast one (preferably in the microsecond speed range) and if the "borrowing" is organized automatically whenever it is required so that the programmer need not be aware of it. One possible way of arranging this is as follows.

The magnetic tape unit has two single word registers associated with it. One of these is a shifting register and is used to assemble a complete word from the characters as they come off the magnetic tape. When a word is complete it is transferred to the second register and an "interrupt program" signal is sent to the central computer. This causes the computer to stop obeying the main program and enter a special sequence of orders to deal with the word from tape; at the same time it records the place where it left the main program. The special sequence of orders checks the parity of the word from magnetic tape, adds it to the sum check if necessary and stores it in the appropriate part of the main store. It then advances the count or index register. If the end of the block has not yet been reached, the computer then returns to the main program. At the end of the block the appropriate checking is done before returning to the main programme.

This plan leaves several difficulties unresolved; it makes, for example, no provision for lock-out during transfers. Another important difficulty is the possibility of several "interrupt program" signals coming at the same time (from different units) or of one special sequence of orders being interrupted by another (or even worse, by itself).

A fairly straightforward way out of these difficulties is to divide all possible interrupt signals into a small number of classes. The classes are allotted in order of priority, and any interrupt signal can only interrupt the program immediately if this is of a lower priority class. If the program being executed is of the same or higher priority class, the interrupt signal is merely stored. At the end of a sequence of orders of one priority, the machine scans the waiting list and starts to operate on the call with the highest remaining priority.

The top priority class would contain operations which cannot afford to wait e.g. transfer of words to or from magnetic tape. The lowest priority would be the main program. There might have to be one or two intermediate classes, the principle of classification being that those processes which have inertia must have a higher priority than those which have none.

It is interesting to consider the timing problems in somewhat more detail. Suppose, for example, that the rate of transfer of information from magnetic tape is one word in 200 μs and that the length of time required for the sequence which stores this word is 5 μs. It may well be that the word is only available for part of interval between successive words, so that, for instance, it must be used within 150 μs of its first appearance. It is clear then, that the maximum number of tape units which could be in operation at once is 30 and that these would use 75% of the total available computer time; the remaining 25% would always be available for lower priority programs. Suppose, however, that we do not want to have a shifting register with each magnetic tape unit, but wish to use the central computer to assemble each word as it comes from the magnetic tape character by character. In this case we can assume that the character interval is 30 μs; we will assume no buffer store so that the character is only available for a short time—say 10 μs. The sequence of orders required to assemble a single word is quite short, but the partly assembled word would have to be stored; it seems reasonable to assume a time of 2 μs for this operation. With these figures a maximum of 5 magnetic tape units could be in use at the same time and these would use 33% of the total machine time. However we still have to put the completed word away. If this takes 5 μs as in the earlier example and we leave the end of word sequence in the same priority class as the character sequence, we shall have to limit the number of magnetic tape units to two. If we can relegate the end of word sequence to a lower priority class, so that it can be interrupted by other character assembly sequences, there is plenty of time to fit it in so that it does not restrict the number of magnetic tape units.

Once more I must emphasise that the examples given here are deliberately simplified. There are a number of other considerations which must be taken into account when calculating timings. For example some operations, such as multiplication and division may be relatively slow. If an interrupt call occurs during one of these the operation, in all probability, will have to be completed. It depends on the details of the logical design of the machine whether this time has to be added to the maximum time for each magnetic tape sequence or whether the relatively simple operations required to deal with magnetic tape can take place while the multiplication or division is being completed. Then again some of the high priority programs will need to use the arithmetic unit; once more depending on the characteristics of the machine it may be necessary to store the contents of various arithmetic and index registers before proceeding with the new program. If this is necessary the general rule should be to store them in locations which are associated with the interrupting program and to restore them again at the end of that program. The amount of information to be stored is determined by what use the interrupting program makes of these common registers. In this way no time will be wasted storing more information than is necessary.

It will be evident that the system of time sharing so far described is quite complex in its operation. I do not know of any already existing computer which uses a system of this sort, though I believe that there are machines under construction which will have time sharing facilities on this scale. It seems to me however, that the scheme described above does not go far enough, and that for a relatively small increase in hardware a very considerably better system can be obtained. The crucial point comes, I think, when we consider the problems of operating a very fast computer.

4. Operating problems

The increasing speed of computers will make the problem of keeping them busy considerably more complicated. Although the major reason for building very large and fast machines may be the existence of problems which cannot be done without them, I think there will always remain far more rather smaller problems. As I have already pointed out, fast machines are much cheaper per operation provided they can be used efficiently. A machine in the microsecond class will be one thousand times as fast as a machine in the millisecond class, but it is not likely to cost more than fifty times as much. We have a factor of twenty to save if we can use it.

The great difficulty, of course, is to keep a very fast machine continuously busy on problems which, for it, are very small. By other standards, of course, these problems are not so small: a problem which involves inverting twelve 20×20 matrices or solving a set of 65 linear equations would not be called small by most standards. This sort of problem would take 30 minutes to an hour on a machine such as Pegasus or the IBM 650; one or two minutes on a Mercury or an IBM 709 and one or two seconds on a microsecond class of machine.

Keeping a steady flow of problems to the machine is obviously going to present very great difficulties and I do not think it will be possible not to lose a few seconds between each problem. It seems worthwhile considering time sharing between operators so that if one operator is idling another may be using the machine. Another possibility is to have a "base load" program in the machine the whole time. This would be a large program which would take a long time to run. It would have the lowest priority and come into operation whenever there was a gap between two shorter programmes.

There are at least two other activities in which a computer is, at present, very inefficiently used. The first of these is

programme checking. For many purposes the best method of programme checking is for a skilled programmer to sit at the operating console of the machine and to plan his operations according to the results produced by the machine. Unfortunately this method is so grossly wasteful of machine time, even with relatively slow machines, that it is generally not allowed except for a few very special problems. The concept of time sharing between operators makes it possible once more to allow this manual program checking at a special console, without seriously interfering with the amount of machine time available for ordinary computing.

The other activity which makes very inefficient use of a computer is the maintenance and adjustment of the peripheral equipment such as paper tape readers and magnetic tape units. Some of these need a considerable amount of adjustment which can only be done satisfactorily by using the computer. If this part of the maintenance can be carried out on a time-sharing basis, it should be possible to reduce the total machine time used for maintenance quite considerably.

Several new problems appear as soon as it becomes possible to have several variable programmes in the machine at the same time. The most important of these is the necessity of seeing that the programmes do not interfere with each other. This is particularly important, of course, if one of the programmes concerned is still under development and so is liable to "run wild." The solution to this difficulty is to provide for interlocks on the main store so that each program is restricted to altering (and perhaps also to reading) numbers in its own section of the store. This in its turn introduces the problem of altering the interlocks. It is evident that it must be possible to change them when required (or it would be impossible to use the whole machine on a single large problem) and for reasons of speed it is obviously desirable to have them altered by a machine instruction. The problem is to ensure that even if a program runs wild so that it obeys a completely unpredictable series of orders, it still shall not be able to alter the interlocks and spoil another programm.

The other rather difficult problems are concerned with the best method of program checking on a machine of this sort. The majority of programmes (and programmers) are not suitable for the manual checking methods. It is therefore necessary to make some provision for other methods of program checking, and it is likely that there will be a considerable amount of this work. A particular problem which arises in this context is the difficulty of determining when a program under test is in error and has come to a loop stop. If this is not detected rapidly, it can waste a disproportionate amount of computer time.

5. The Director

The general organization of the computer may now appear to be formidably complicated. If it were necessary to design and build special hardware to perform all these functions, it would be a somewhat daunting task. Fortunately, however, it is not necessary to do this. The computer itself is designed to deal with just such complicated logical problems, and viewed as problems for a computer programme, the organization of the machine is by no means unusally difficult or complicated. We therefore seek to control the organization of the machine by means of a programme which I shall call the Director.

The first important characteristic of the Director is that it is a fixed program. This means that it can, and should, be kept in a special part of the store which is non-erasable. It is a fortunate fact that it is relatively cheap and easy to provide a non-erasable store with a very fast read-out time (of the order of 1/5 μs). This reduces the time spent in the Director and ensures that the program of the Director can never be destroyed.

The second fact is that the Director requires some working space which is immune from interference by other programmes. That is to say that we require some ordinary erasable store which can only be altered by instructions in the Director.

The fact that the Director is in a special, fixed part of the store makes it possible to introduce machine instructions which are only accessible to the Director. If the instructions which alter the main store interlocks are of this sort, it is quite easy to arrange that no program can alter the interlocks of any other program, and thus to give protection against programmes under test running wild.

6. Hypothetical machine

In order to clarify and give precision to the ideas discussed so far, I propose to describe, in outline, a possible design for a large fast computer making extensive use of time sharing. I must make it quite clear that the machine I am going to describe is not, so far as I know, an actual or projected computer. The performance figures I use have no secret significance, they are merely estimates of what I consider reasonable either now or in the near future. In fact the main characteristics of the machine would not be critically dependent on the actual speeds assumed.

6.1 *Characteristics of the machine*

The machine is intended for mathematical work. It is a binary machine with both fixed and floating point arithmetic. The word length is between 40 and 50 bits. It has a main store of 32,000 words or more (on ferrite cores) with a cycle time of 1 to 2 μs. It also has a fixed (non-erasable) store of 1000 to 4000 words with a read-out time of $^1/_5$ μs. There is a special working store of a few hundred words which is only accessible from instructions in the fixed store. Arithmetic operations (addition, multiplication, division, fixed or floating point) take from 2 to 10 μs, including access time. Red tape instructions reqnire 1 to 2 μs if obeyed from the main store and from $^1/_4$ to $^1/_2$ μs if obeyed from the fixed store.

The details of the order structure and the provision of modifier registers (index registers) will not be fully discussed. The number of modifier registers available will influence the method used to change from one programme to another, and in particular, if the number is very small (say two or three) it may be desirable to have special loading and dumping instructions for them.

The backing store for the machine will be magnetic tape using fixed block lengths. The transfer rate will be 20,000 to 60,000 6-bit characters per second which is equivalent to one word every 100 to 300 μs. Normally speaking, each magnetic tape unit will have two one-word buffers associated with it as described in section 3 above. There will be 8 to 10 units associated with the main computer.

6.2 *Input and output equipment*

There will be up to 20 slow input/output stations. Each consists of a paper tape reader (200 characters/sec), a paper tape punch and printer (or an electric typewriter) and a magnetic tape mechanism, together with a working space and a few control keys. The magnetic tape mechanism can be considerably simpler than the ones associated with the main machine as it is not required to work nearly so fast. In addition there will be fast input/output stations viz: a punched card input and output, a line printer (or perhaps two) and probably a graphical display output. Each of these will have an associated magnetic tape unit.

6.3 *Operator's consoles*

There will be three operator's consoles: an engineer's console for maintaining and testing the machine, a programme-

testing console with various visual displays and at least an electric typewriter, and a main operator's console. All three will have fairly extensive facilities for intervening in the course of a calculation.
In addition each of the input/output stations will have sufficient control keys to allow them to be operated as if they were off-line mechanisms.

6.4 *Method of operation*

In normal operation there would be time sharing among up to four variable programmes.

I) A 'base load' program. This is a long term program which has already been checked. It is set up initially from the main operator's console and given the lowest priority. It is used to fill in the gaps between other programmes.

II) A series of short run programmes. These are run in sequence from the main operator's console. They constitute the majority of short and medium length programmes. In general they will have been written with the aid of some form of automatic coding routine and will have been translated into machine code and stored in sequence on magnetic tape in a form suitable for the input routine.

III) A programme being manually checked at the programme-testing console.

IV) An engineer's test routine being used to service part of the peripheral equipment.

In addition to these a number of the slow input/output stations will be in use. These have three main functions.

I) The initial input of program (and data) would be from punched paper tape. This would be read in from one of the stations and immediately stored (probably without much translation) on magnetic tape. At some later time a special translating run would be initiated by the operator at the main console and the resulting programmes stored in a single magnetic tape for later running.

II) The stations would also be used for printing the result of programmes whose volume of output did not warrant the use of the line printer. These results would be stored by the main computer on the magnetic tape of one of the input/output stations and later printed out.

III) The third use of the stations would be for post mortem printing when testing a program which did not require manual testing. When a program being tested in this way was finished, or if it broke down in any detectable way or if it had been running for longer than was forecast, the contents of various prearranged parts of the store (probably the whole store used by the problem) would be output (by the Director) onto magnetic tape. It would then be possible for the programmer at some later time to print out any parts of the store he required by using a punched paper steering tape at one of the input/output stations. If after further thought he needed more information, a further steering tape would provide it.

This illustrates one way in which the difficulty of loop stops can be dealt with. It would be mandatory for the programmer to supply a time-estimate for his program. (For standard programmes, such as matrix inversions, which were data-dependent, this would be calculated by an interlude on input. If the programmer omitted to give an estimate, a standard one of one second could be used.) The Director would keep a record of the length of time spent in each program, and if the estimate were exceeded by, say, a factor of two, treat it as a program error.
In the normal course of events the line printer and other fast input/output devices would be used with their associated magnetic tape units as if they were off-line.

In this way, during the normal running of the machine several operators are using the machine during the same time. To each of these operators the machine appears to behave as a separate machine (smaller, of course, and slower than the real machine). Nevertheless whenever it becomes desirable the entire machine can be devoted to a single large scale problem.

7. Interlock problems

There are one or two difficulties connected with interlocks which were mentioned above but left without any very precise solution. It is now possible to suggest rather more detailed solutions for them in the framework of the hypothetical machine.
To prevent one variable program interfering with another, each program is required to be compact in the store. (This is quite easily arranged if automatic coding is used.) On input the Director assigns an unused block of store to the problem, inserts the relevant addresses in the instructions and establishes the store limits inside which the problem must lie. While the problem is being run these limits are held in two special 'limit registers' and are compared with the address register of the main store every time this is used. If the address called for is not in the correct part of the program a program error is indicated to the Director. This comparison need not slow down the process of reading from store. It can proceed in parallel with the read out, and if the address is out of limits, the same number is merely written back into the store again.
The instructions setting the limit registers are only available to the fixed store, and it is part of the function of the Director to set them correctly whenever control is moved to a fresh program. The allocation of store is thus a function of the Director. Any program can remove itself (for example when it comes to an end) but no variable program can remove another. Only the Director, can forcibly remove a program, and it will only do this if there is a program error (e.g. a call for a number outside its limits) or if the program overruns its time limit.
When fixed programmes (such as those for magnetic tape) are required to use part of the store belonging to a variable program, they (or the Director) will check on first entry that the part of the store they are being asked to use is a permissible one. It is also desirable to provide a temporary lock-out while transferring blocks of data to and from magnetic tape (see section 3). This can be done rather neatly if there is a parity check bit on words in the main store. The suggestion is that the parity bit should be reversed for the period when it is necessary to lock out the main computer (but not the magnetic tape fixed programme).This involves providing instructions (accessible only from the fixed store) for reading and writing in the main store with reversed parity. An attempt by the main program to read a word with reversed parity would lead (as all error indications would lead) to a sequence in the Director. This would determine whether a magnetic transfer involving that address was in process. If it were, it would delay the main program; if not it would signal a store error.

8. Conclusion

The overall aim of the system which has been described above is to separate the various operations which are time sharing, as much as possible, while still preserving the possibility of running the whole machine as a single unit. The advantages which follow are firstly a cheaper machine because the peripheral equipment is simpler, secondly a more efficient use of the machine, because it is more continuous and lastly, considerably greater flexibility, because various time-sharing schemes can be tried without very great expense by merely rewriting the Director.

9. Appendix

9.1 *Functions of the Director*

a) Priority shifting sequences
- I) On moving to higher priority.
  - Store present control number and priority.
  - Copy accumulators, modifiers etc., as required for interrupting program. The storage registers used are allocated to the interrupting program.
  - Alter store address limit registers.
  - Enter interrupting program.
- II) On moving to lower priority (i.e. at 'wait' order).
  - Restore accumulators, modifiers etc., control no., store address limit register.
  - Prepare to enter program. If another interrupt of higher priority is still in force, enter the appropriate sequence instead.

b) Arrange allocation of store during input and freeing of store at end of program.

c) Keep time for programmes being checked (and possibly others).

d) If unexpected overflows, or forbidden store references, or non-existant orders or other incorrect instructions occur, treat as a program error.

e) Special operating for machine errors and power failures.

f) Monitoring printing at main console.

9.2 *Orders only available to fixed store*

- Access to special working store.
- Set limit registers.
- Read and write in main store with reversed parity.
- Operate peripheral equipment.
- Operate mechanical interlocks on magnetic tape units.

9.3 *Tentative priority list for Director*

High priority	Power failure
	Machine breakdown
	Director priority changing sequences
	Magnetic tape read/write
	Line printer
	Paper tape input
	Monitoring printing
	Output and post mortem printing
	Other Director functions
	Engineers' testing
	Manual program testing
	Short run programmes
Low priority	Base load program.

It may be desirable to divide these into various groups, or even to allow the priority of a program to be decided by the Director.

10. Discussion

E. F. Codd (USA): First, I wish to report that an approach very similar to that described by Mr. Strachey has been taken on the IBM STRETCH computer. Special provisions have been made in the STRETCH computer to facilitate multi-programming. Development of a supervisory program intended to supervise several problem programs executed concurrently was begun before the logical organization of STRETCH was finally determined. We hope to report our progress in multi-programming on STRETCH at the forthcoming meeting of the Eastern Joint Computer Conference. One of the most difficult problems is that of determining in a sufficiently general way, the best sequence in which problem programs should be brought into the execution phase. Associated with this problem is that of storage allocation. Does Mr. Strachey have any solution (not of an ad hoc nature) to these problems?

C. Strachey: I have no solution.

P. Dreyfus (France): I disagree on the possibility of handling time sharing by a *program*. In my opinion, it must be handled by *hardware*. The Director works for each instruction and each transfer. A program is too slow to perform these functions.
If the Director operators are very fast, it means that the machine could be very fast so that the relative loss in time is the same: probably a 10/1 reduction over basic speed. Whereas hardware handling of time sharing results in a speed reduction of only 1.5/1.
The cost of integrating the Director in the hardware of the machine represents less than a 20% increase in the overall cost of the equipment.

C. Strachey: It is not necessary to bring in the Director after each instruction, it is only called in at the request of a piece of peripheral equipment, and whenever the program operating at the moment has to wait for any reason. These interruptions should be relatively infrequent. The practicability of using a program contested time sharing scheme, certainly depends on the rate of interruption being quite small compared with the rate at which instructions are obeyed, but as the interruption rate depends generally on mechanical considerations, it is unlikely to increase in the future as much as machine speeds.

J. Porte (France): For scientific research, there is a question which seems to have been neglected—not only by Mr. Strachey, but by other speakers at today's meeting, including Mr. Wilkes. The question is that of time-sharing in programming versus time-sharing in calculation.
For much special or experimental research, we cannot hope that programs will be generated by "auto-programming" methods—at least in the near future. Consequently, we must write programs in machine language, and that means that time will be lost in the checking of programs.
As far as I can see at the present time, the result of such considerations is to emphasize the usefulness of a wide range of order codes, particularly double- or triple-length arithmetic and double- or triple-length floating point arithmetic, for experience shows that in such experimental work we are often faced with a lack of accuracy.

C. Strachey: The checking of programs which are so large that they occupy the whole machine will inevitably be an expensive task and would almost certainly not be done directly from the machine console as it could not be time-shared with any other program. Very large programs would probably be checked in sections as far as possible to avoid this difficulty.
As for experimental programs requiring multiple precision arithmetic, it seems to me to be a reasonable request that automatic programming systems for very large and fast machines should include these facilities as a matter of course.